Generating and matching hashes of multimedia content.

FIELD OF THE INVENTION

The invention relates to a method and arrangement for generating a hash signal identifying an information signal. The invention also relates to a method and arrangement for matching such a hash signal with hash signals stored in a database.

BACKGROUND OF THE INVENTION

Hash functions are generally known in the field of cryptography, where they are used, inter alia, to identify large amounts of data. For instance, in order to verify correct reception of a large file, it suffices to send the hash value (also referred to as signature) of that file. If the returned hash value matches the hash value of the original file, there is almost complete certainty that the file has been correctly received by the receiving party. The remaining uncertainty is introduced due to the fact that a collision might occur: i.e. two different files may have the same hash value. A carefully designed hash function minimizes the probability of collision.

A particular property of a cryptographic hash is its extreme fragility. Flipping a single bit in the source data will generally result in a completely different hash value. This makes cryptographic hashing unsuitable for identifying multimedia content where different quality versions of the same content should yield the same signature. Signatures of multimedia content that are to a certain extent invariant to data processing (as long as the processing retains an acceptable quality of the content) are referred to as robust signatures or, which is our preferred naming convention, robust hashes. By using a database of robust hashes and content identifiers, unknown content can be identified, even if it is degraded (e.g. by compression or AD/DA conversion). Robust hashes capture the perceptually essential parts of audio-visual content.

Using a robust hash to identify multimedia content is an alternative to using watermarking technology for the same purpose. There is, however, also a great difference. Whereas watermarking requires action on original content (viz. watermark embedding) before being released, with its potential impact on content quality and logistical problems, robust hashing requires no action before release. The drawback of hashing technology is that

access to a database is needed (e.g. hashing is only viable in a connected context), whereas watermark detectors can operate locally (for example in non-connected DVD players).

United States Patent 4,677,466 discloses a known method of deriving a signature from a television signal for broadcast monitoring. In this prior art method, the signature is derived from a short video or audio sequence after the occurrence of a specified event such as a blank frame.

OBJECT AND SUMMARY OF THE INVENTION

It is a general object of the invention to provide a robust hashing technology. More particularly, it is a first object of the invention to provide a method and arrangement for extracting a limited number of hashing bits from multimedia content. The hashing bits are robust, but not in a sense that the probability of bit errors is zero. It is known that non-exact pattern matching (i.e. searching for the most similar hash value in the database) is NP-complete. In layman's terms, this means that the best search strategy is an exhaustive search, which is prohibitive in many applications dealing with large databases. Therefore, a second object of the invention is to provide a method and arrangement that overcomes this NP-complete search complexity.

The first object is achieved by dividing the information signal into successive (preferably overlapping) frames, computing a hash word for each frame, and concatenating successive hash words to constitute a hash signal (or hash in short). The hash word is computed by thresholding a scalar property or a vector of properties of the information signal, for example, the energy of disjoint frequency bands or the mean luminance of image blocks.

The second object is achieved by selecting a single hash word of an input block of hash words, searching said hash word in the database, calculating a difference between the input block of hash words and a corresponding stored block of hash words. These steps are repeated for further selected hash words until said difference is lower than a predetermined threshold.

Further features of the invention are defined in the subclaims.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic diagram of an embodiment of an arrangement for extracting a hash signal from an audio signal in accordance with the invention.

Fig. 2 is a diagram illustrating the subdivision of an audio signal spectrum into logarithmically spaced bands.

Fig. 3 is a diagram illustrating hash words extracted from an audio clip.

Fig. 4 is a schematic diagram of an embodiment of an arrangement for extracting a hash signal from a video signal in accordance with the invention.

Fig. 5 is a diagram illustrating hash words extracted from a video sequence.

Fig. 6 is a flow chart of operations carried out by a computer which is shown in Fig. 1 in accordance with the invention.

Fig. 7 is a diagram to illustrate the operation of a computer which is shown in Fig. 1.

Fig. 8 shows a graph of the number or bit errors in hash words forming an extracted hash block which is shown in Fig. 3.

Fig. 9 shows a graph of the most reliable bit of the hash words of the hash block which is shown in Fig. 3.

Fig. 10 is a flow chart of operations carried out by the computer which is shown in Fig. 1 in accordance with a further embodiment of the invention.

DESCRIPTION OF EMBODIMENTS

Before describing a preferred embodiment, a general description of considerations underlying this invention will be elucidated.

Two signals (audio, video, image) can differ quite drastically (e.g. by compression) in a signal theoretical sense, whereas they are perceptually indistinguishable. Ideally, a hash function mimics the behavior of the human auditory system (HAS) or human visual system (HVS), i.e. it produces the same hash signal for content that is considered the same by the HAS/HVS. However, many kinds of processing (compression, noise addition, echo addition, D/A and A/D conversion, equalization etc.) can be applied to the signal and there is no algorithm that is able to mimic the HAS/HVS perfectly. A complicating factor is that even the HAS/HVS varies from person to person as well as in time, and even the notion of one single HAS/HVS is untenable. Also, the classical definition of a hash does not take time into account: a robust hash should not only be able to identify the content, but should also be able to identify time (intervals). For this reason the following definition for a robust hash is herein used: A robust hash is a function that associates with every basic time-unit of multimedia content a semi-unique bit-sequence that is continuous with respect to content similarity as perceived by the HAS/HVS.

In other words, if the HAS/HVS identifies two pieces of audio, video or image as being very similar, the associated hashes should also be very similar. In particular, the hashes of original content and compressed content should be similar. Also, if hash words are computed for overlapping frames, the hash words should be similar, i.e. hashes should have a low pass character. On the other hand, if two signals really represent different content, the robust hash should be able to distinguish the two signals (semi-unique). This is similar to the collision requirement for classical cryptographic hashes. The required robustness of the hashing function is achieved by deriving the hash function from robust features (properties), i.e. features that are to a large degree invariant to processing. Robustness can be expressed by the Bit Error Rate (BER), which is defined as the ratio of the number of erroneous bits and the total number of bits.

Robust hashing enables content identification which is the basis for many interesting applications. Consider the example of identification of content in a multimedia database. Suppose one is viewing a scene from a movie and would like to know from which movie the shot originates. One way of finding out is by comparing the scene to all fragments of the same size of all movies in the database. Obviously, this is totally infeasible in case of a large database: even a short video scene is represented by a large amount of bytes and potentially these have to be compared to the whole database. Thus, for this to work, one needs to store a large amount of easily accessible data and all these data have to be compared with the video scene to be identified. Therefore, there is both a storage problem (the database) as well as a computational problem (matching large amounts of data). Robust hashing alleviates both problems by reducing the number of bits needed to represent the video scenes: fewer bits need to be stored and fewer bits need to be used in the comparison.

Robust hashing of audio signals will be described first. The audio signal will be assumed to be mono audio that has been sampled at a sample frequency of 44.1 kHz (CD-quality). If the audio is stereo, there are two options: either hash signals are extracted for the left and the right channel separately, or the left and the right channel are added prior to hash signal extraction.

Even if we only have a short piece of audio (of the order of seconds), we would like to determine which song it is. As audio can be seen as an endless stream of audio-samples, it is necessary to subdivide audio signals into time intervals or frames and to calculate a hash word for every frame.

Very often, when trying to match hashes in a database, it is impossible to determine the frame boundaries. This synchronization problem is particularly applicable to

audio hashing. This problem is solved by dividing the signal into overlapping frames. Overlapping also ensures that hash words of contiguous frames have a certain amount of correlation. In other words, the hashes change slowly over time.

Fig. 1 shows a schematic diagram of an embodiment of an arrangement for generating an audio hash signal in accordance with the invention. The audio signal is first downsampled in a downsampler **11** to reduce the complexity of subsequent operations and restrict the operation to a frequency range of 300-3000 Hz, which is most relevant for the Human Auditory System.

In a framing circuit **12**, the audio signal is divided into frames. The frames are weighed by a Hanning window having a length of 16384 samples (≈0.4 seconds) and an overlap factor of 31/32. The overlap is chosen in such a way that a high correlation of the hash words between subsequent frames is ensured. The spectral representation of every frame is computed by a Fourier transform circuit **13**. In the next block **14**, the absolute values (magnitudes) of the (complex) Fourier coefficients are computed.

A band division stage **15** divides the frequency spectrum into a number (e.g. 33) of bands. In Fig. 1, this is schematically shown by selectors **151**, each of which selects the Fourier coefficients of the respective band. In a preferred embodiment of the arrangement, the bands have a logarithmic spacing, because the HAS also operates on approximately logarithmic bands. By choosing the bands in this manner, the hash will be less susceptible to processing changes such as compression and filtering. In the preferred embodiment, the first band starts at 300Hz and every band has a bandwidth of one musical tone (i.e. the bandwidth increases by a factor of $2^{1/12} \approx 1.06$ per band). Fig. 2 shows an example of a spectrum **201** of a frame and the subdivision thereof into logarithmically spaced bands **202**.

Subsequently, for every band a certain (not necessarily scalar) characteristic property is calculated. Examples of properties are energy, tonality and standard deviation of the power spectral density. In general, the chosen property can be an arbitrary function of the Fourier coefficients. Experimentally it has been verified that the energy of every band is a property that is most robust to many kinds of processing. This energy computation is carried out in an energy computing stage **16**. For each band, it comprises a stage **161** which computes the sum of the (squared) magnitudes of the Fourier coefficients within that band.

In order to get a binary hash word for each frame, the robust properties are subsequently converted into bits. The bits can be assigned by calculating an arbitrary function of the robust properties of possibly different frames and then comparing it to a threshold

value. The threshold itself might also be a result of another function of the robust property values.

In the present arrangement, a bit derivation circuit **17** converts the energy levels of the bands into a binary hash word. In a simple embodiment, the bit derivation stage generates one bit for each band, for example, a '1' if the energy level is above a threshold and a '0' if the energy level is below said threshold. The thresholds may vary from band to band. Alternatively, a band is assigned a hash bit '1' if its energy level is larger than the energy level of its neighbor, otherwise the hash bit is '0'. The present embodiment uses an even improved version of the latter alternative. To prevent a major single frequency in the audio signal from producing identical hash words for successive frames, variations of the amplitude over time are also taken into account. More particularly, a band is assigned a hash bit '1' if its energy level is larger than the energy level of its neighbor and if that was also the case in the previous frame, otherwise the hash bit is '0'. If we denote the energy of a band m of frame n by EB(n,m) and the m-th bit of the hash word H of frame n by H(n,m), the bit derivation circuit **17** generates the bits of the hash word in the following manner:

$$H(n,m)=\begin{cases}1 & \text{if } EB(n,m)-EB(n,m+1)-(EB(n-1,m)-EB(n-1,m+1))>0\\ 0 & \text{if } EB(n,m)-EB(n,m+1)-(EB(n-1,m)-EB(n-1,m+1))\le 0\end{cases}$$

To this end, the bit derivation circuit **17** comprises, for each band, a first subtractor **171**, a frame delay **172**, a second subtractor **173**, and a comparator **174**. The 33 energy levels of the spectrum of an audio frame are thus converted into a 32-bit hash word. The hash words of successive frames are finally stored in a buffer **18**, which is accessible by a computer **20**. The computer stores the robust hashes of a large number of original songs in a database **21**.

In a subsequent operation, the same arrangement computes the hash of an unknown audio clip. Reference numeral **31** in Fig. 3 shows the hash words of 256 successive overlapping audio frames (≈3 seconds) of the audio clip as stored in the database **21**. In the Figure, each row is a 32-bit hash word, a white pixel represents a '1' bit of the hash word, a black pixel represents a '0' bit, and time proceeds from top to bottom. Reference numeral **32** shows the hash words extracted from the same audio clip after MP3 compression at 32 kBit/s. Ideally, the two hash blocks should be identical, but due to the compression some bits are different. The difference is denoted **33** in Fig. 3.

Robust hashing of image or video signals will now be described. Again, the robust hashes are derived from specific features of the information signal. The first question to be asked is in which domain to extract said features which determine the hash word. In

contrast to audio, where the frequency domain optimally represents the perceptual characteristics, it is less clear which domain to use. For complexity reasons it is preferable to avoid complex operations, like DCT or DFT transformations. Therefore, features in the spatio-temporal domain are computed. Moreover, to allow easy feature extraction from most compressed video streams as well, features are chosen which can be easily computed from block-based DCT coefficients.

Based on these considerations, the preferred algorithm is based on simple statistics, like mean and variance, computed over relatively large image regions. The regions are chosen in a fairly simple way: the image frame is divided into square blocks of 64 by 64 pixels. The features are extracted from the luminance component. This is, however, not a fundamental choice: the chrominance components may be used, as well. As a matter of fact, the easiest way to increase the number of hash bits is to extract them from the chrominance components in a similar way as the extraction from the luminance.

Fig. 4 shows a block diagram of an arrangement for generating a hash signal identifying a video signal in accordance with the invention. The arrangement receives successive frames of the video signal. Each frame is divided (**41**) in M+1 blocks. For each of these blocks, the mean of the luminance values of the pixels is computed (**42**). The mean luminance of block k in frame p is denoted F(p,k) for k=0,...,M.

In order to make the hash independent of the global level and scale of the luminance, the luminance differences between two consecutive blocks are computed (**43**). Moreover, in order to reduce the correlation of the hash words in the temporal direction, the difference of spatial differential mean luminance values in consecutive frames is also computed (**44, 45**). In other words, a simple spatio-temporal 2×2 Haar filter is applied to the mean luminance. The sign of the result constitutes (**46**) the hash bit H(p,k) for block k in frame p. In mathematical notation:

$$H(p,k)=\begin{cases}1 & \text{if } (F(p,k)-F(p,k-1))-(F(p-1,k)-F(p-1,k-1))\geq 0\\ 0 & \text{if } (F(p,k)-F(p,k-1))-(F(p-1,k)-F(p-1,k-1))<0\end{cases}$$

In this example, each frame is divided in 33 blocks (i.e., M=32) of size 64×64. A complete hash H consists of the bits extracted from 30 consecutive frames. Such a hash block, consisting of 30 hash words of 32 bits each (960 bits) leads to a sufficiently small false positive probability, as will be shown below. A typical original hash block is depicted **51** in Fig. 5, where black and white correspond to '0' and '1', respectively. The corresponding hash block of the same material scaled horizontally to 94% is denoted by reference numeral **52.** Numeral **53** denotes the difference between the hash blocks **51** and **52.** In this case the bit

error rate equals 11.3%. Note how indeed the erroneous bits have a strong correlation in the temporal (vertical) direction.

The process of matching extracted hash blocks to the hash blocks in a large database will now be described. This is a non-trivial task since it is well-known that imperfect matching (remember that the extracted hash words may have bit errors) is NP-complete. This will be shown by means of the following (audio) example. In a database, 100,000 songs of approximately five minutes (≡25000 hash words per song) are stored. It will be assumed that a hash block having 256 hash words (e.g. hash block **32** in Fig. 3) has been extracted from the unknown audio clip. It is now to be determined to which of the 100,000 stored songs the extracted hash block matches best. Hence the position of a hash block in one of the 100,000 songs has to be found, which most resembles the extracted hash block, i.e. for which the bit error rate (BER) is minimal or, alternatively, for which the BER is lower than a certain threshold. The threshold directly determines the false positive rate, i.e. the rate at which songs are incorrectly identified from the database.

Two 3 seconds audio clips (or two 30-frame video sequences) are declared similar if the Hamming distance between the two derived hash blocks H_1 and H_2 is below a certain threshold T. This threshold T directly determines the false positive rate P_f, i.e. the rate at which two audio clips / video sequences are incorrectly declared equal (i.e. incorrectly in the eyes of a human beholder): the smaller T, the smaller the probability P_f will be. On the other hand, a small value T will negatively effect the false negative probability P_n, i.e. the probability that two signals are 'equal', but not identified as such. In order to analyze the choice of this threshold T, we assume that the hash extraction process yields random i.i.d. (independent and identically distributed) bits. The number of bit errors will then have a binomial distribution with parameters (n,p), where n equals the number of bits extracted and p (=0.5) is the probability that a '0' or '1' bit is extracted. Since n (32×256=8192 for audio, 32×30=960 for video) is large in our application, the binomial distribution can be approximated by a normal distribution with a mean $\mu=np$ and standard deviation $\sigma = \sqrt{np(1-p)}$. Given a hash block H_1, the probability that a randomly selected hash block H_2 has less than $T=\alpha n$ errors with respect to H_1 is given by:

$$P_f(\alpha) = \frac{1}{2\pi} \int_{(1-2\alpha)\sqrt{n}}^{\infty} e^{-\frac{x^2}{2}} dx = \frac{1}{2} \operatorname{erfc}\left(\frac{1-2\alpha}{\sqrt{2}} \sqrt{n} \right) \qquad (1)$$

However, in practice the robust hashes have high correlation along the time axis. This is due to the large time correlation of the underlying video sequence, or the overlap

of audio frames. Experiments show that the number of erroneous bits are normally distributed, but that the standard deviation is approximately 3/2 times larger than the i.i.d. case. Equation (1) therefore is modified to include this factor 3/2.

$$P_f(\alpha) = \frac{1}{2}\mathrm{erfc}\left(\frac{1-2\alpha}{3}\sqrt{2n}\right) \qquad (2)$$

The threshold for the BER used during experiments was α=0.25. This means that, of 8192 bits, less than 2048 bit errors have to occur in order to decide that the hash block originates from the same song. In this case the bit errors have a normal distribution with a mean μ of np=4096 and a standard deviation σ of $3\sqrt{(np(1-p))}$=135.76. The chosen threshold setting then corresponds to the false alarm probability of 15.2σ. Hence, the false alarm probability equals $1.8\cdot10^{-52}$. Note, however, that the false alarm probability will be higher in practice if music with similar hash words (e.g. a Mozart piece played by two different pianists) is included in the database.

Searching the position of the extracted hash block in the database can be done by brute force matching. This will take around 2.5 billion (=25000×100,000) matches. Moreover, the number of matches increases linearly with the size of the database.

In accordance with an aspect of the invention, the computer **20** uses a more efficient strategy for finding the corresponding song in the database **21**. Fig. 6 is a flow chart of operations carried out by the computer. Upon storing an original song in the database, the computer updates a lookup table (LUT) in a step **60**. The LUT is shown as a separate memory **22** in Fig. 1, but it will be appreciated that it will be part of the large database memory **21** in practice. As is shown in Fig. 7, the LUT **22** has an entry for each possible 32-bit hash word. Each entry of the LUT points to the song(s) and the position(s) in that song where the respective hash word occurs. Since a hash word can occur at multiple positions in multiple songs, the song pointers are stored in a linked list. Thus the LUT can generate multiple candidate songs. Note that a LUT containing 2^{32} entries can be impractical when there is only a limited number of songs in the database. In such a case, it is advantageous to implement the LUT with a hash table and a linked list. Reference numeral **70** in Fig. 7 denotes a block of 256 hash words extracted from the unknown audio clip (e.g. hash block **32** in Fig. 3).

In a first embodiment of the matching method, it will be assumed that every now and then a single hash word has no bit errors. In a step **61**, a single hash word H(m) is selected from the hash block and sent to the database. Initially, this will be the last hash word H(256) of the extracted hash block. In the example shown in Fig. 7, this is the hash word

0x00000001. The LUT in the database points to a certain position in song 1. Let it be assumed that this position is position p. In a step **62**, the computer calculates the BER between the extracted hash block and the block of hash words from position p-255 until position p of song 1 (denoted **71** in Fig. 7). In a step **63**, it is checked whether the BER is low (<0.25) or high. If the BER is low, there will be a high probability that the extracted hash words originate from song 1. If the BER is high, either the song is not in the database or the single hash word H(m) contains an error. The latter will be assumed to be the case in this example. Another single hash word is then selected in a step **64** and looked up in the LUT. In Fig. 7, the last but one single hash word H(255) is now being looked up. This hash word appears to occur in song 2. The BER between input block **70** and stored block **72** appears to be lower than 0.25 now, so that song 2 is identified as the song from which the audio clip originates. Note that the last hash word in the stored block **52** is 0x00000000. Apparently, the previously selected hash word 0x0000001 had one bit error.

The computer thus only looks at one single hash word at a time and assumes that every now and then such a single hash word has no bit errors. The BER of the extracted hash block is then compared with the corresponding (on the time axis) hash blocks of the candidate songs. The title of the candidate song with the lowest BER will be chosen as the song from which the extracted hash words originate, provided that the lowest BER is below the threshold (step **65**). Otherwise, the database will report that the extracted hash block was not found. Another single hash word will then be tried. If none of the single hash words leads to success (step **66**), the database will respond by reporting the absence of the candidate song in the database (step **67**).

The above-described method relies on the assumption that every now and then an extracted hash word has no bit errors, i.e. it is perfectly equal to the corresponding stored hash word. Extensive experiments have shown that this occurs regularly a few times per second for most audio. This is shown, for example, in Fig. 8 which shows the number of bit errors in the 256 hash words forming the extracted block of Fig. 3B. Thirteen hash words occur without any bit errors in this 3-second audio clip.

However, it is unlikely that hash words without any bit errors occur when the audio is severely processed. In that case, the title of the song cannot be retrieved by means of the previous method. To this end, another embodiment of the matching method will be described. This method uses soft information of the hash extraction algorithm to find the extracted hash words in the database. Soft information is understood to mean the reliability of a bit, or the probability that a hash bit has been retrieved correctly. In this embodiment, the

arrangement for extracting the hash words includes a bit reliability determining circuit. The bit reliability determining circuit is denoted **19** in the audio hash extraction arrangement which is shown in Fig. 1. It circuit receives the differential energy band levels in the form of real numbers. If the real number is very close to the threshold (which is zero in this example), the respective hash bit is unreliable. If instead the number is very far from the threshold, it is a reliable hash bit. The threshold can be fixed or controlled such that the number of reliable bits is fixed.

The bit reliability determining circuit **19** determines the reliability of every hash bit, and thus enables the extraction arrangement or the computer **20** to generate a list of most probable alternative hash words for each hash word. By assuming again that at least one of the alternative hash words is correct, the song title can be received correctly and easily. Fig. 9 shows, for all the 256 hash words of hash block **32** in Fig. 3, which bit of the hash word is the most reliable.

Fig. 10 is a flow chart of operations carried out by the computer in this embodiment of the method of finding the extracted hash block in the database. The same reference numerals are used for operations already described before. Again, the last extracted hash word (0x00000001, see Fig. 7) of the hash block is initially selected and sent to the database (step **61**). The LUT in the database points to position p in song 1. The BER between the extracted hash block and the corresponding block **71** in song 1 is calculated (step **62**). Meanwhile, it is known from the previous example that the BER is high. In a step **101**, the computer now consults the bit reliability determining circuit **19** (Fig. 1) and learns that bit 0 is the least reliable bit of this particular hash word. The next most probable candidate hash word is now obtained by flipping said bit. The new hash word (0x00000000) is sent to the database in a step **102**. As is shown in Fig. 7, the hash word 0x00000000 leads to two possible candidate songs in the database: song 1 and song 2. If, for example, the extracted hash words now have a low BER with the hash words of song 2, song 2 will be identified as the song from which the extracted hash block originates. Otherwise, new hash word candidates will be generated, or another hash word will be used to try to find the respective song in the database. This strategy is continued until it is found in a step **103** that there are no further alternative candidate hash words.

Note that, once a piece of audio is identified in practice as originating from a certain song, the database can first try to match the extracted hash words with that song before generating all the candidate hash words.

A very simple way of generating a list of most probable hash words is to include all the hash words with N most reliable bits being fixed and every possible combination for the remaining bits. In the case of 32 bits per hash and choosing N=23, a list of 512 candidate hash words is required. Furthermore it means that the 9 least reliable bits of the hash word can be wrong before an audio excerpt cannot be identified anymore. For the case shown in Figure 6, this means that 117 hash words, instead of 13 with the previous method, will yield a correct pointer to the song in the database.

In an alternative embodiment of the matching method, the matching is done only on the basis of hash bits being marked as reliable. This method is based on the insight that it is unnecessary to compare unreliable bits of a received hash with the corresponding bits in the database. This leads to a far smaller bit error rate, although this comes at the cost of a more complicated search strategy and a larger bandwidth needed to transmit all necessary information to the database.

A few applications of robust hashing will now be described.

- Broadcast Monitoring: A broadcast monitoring system consists of two parts: a central database containing the hashes of a large number of songs, and monitoring stations that extract a hash block from the audio that is broadcast by, for instance, radio stations. The monitoring station will send the extracted hash block to the central database and then the database will be able to determine which song has been broadcast.
- Mobile Phone Audio Info: Imagine that you are in a bar and hear a song of which you want to know the title. You then just pick up your mobile telephone and call an audiohash database. The audiohash database will then hear the song and extract a hash block. If it then finds the hash block in the database, it will report back the title of the song.
- Connected Content (MediaBridge): The company Digimarc currently has an application called MediaBridge, which is based on watermarking technology. The idea is that a watermark in a piece of multimedia will direct a user to a certain URL on the Internet where he can get some extra information. E.g. an advertisement in a magazine is watermarked. By holding this advertisement in front of a webcam, a watermark detector will extract a watermark key that is sent to a database. This database then contains the URL to which the user will be redirected. The same application can work with the use of robust hashing technology. In the future, one might even think of a person pointing his mobile videophone at a real-life object. The audio hash database will then report back information about this object, either directly or via an URL on the Internet.

- Multimedia Quality Metering: If the hash words of high quality original content are listed in the database, a quality measure can be obtained by determining the BER of the extracted hash words of processed multimedia content.

From an abstract point of view, the robust audio hashes are derived from an audio signal by comparing energy in different frequency bands and over time. A generalization of this approach is to consider any cascade of LTI and non-linear functions. In particular, a robust hash can also be obtained by applying a (dyadic) filter bank (an LTI operator), followed by squaring or taking absolute words (a non-linear function), followed by a difference operator over time and/or band (an LTI operator), finally followed by a thresholding operator. By applying a carefully designed linear filter bank as an initial operator, the complexity of a FFT can be avoided. Moreover, as many compression engines have a linear filter bank as an initial phase, there is the option to integrate feature extraction with compression.

It is further noted that robust hashing and digital watermarks can be used in combination to identify content. The method described above and some watermark detection algorithms have a number of initial processing steps in common, viz. the computation of the spectral representation. This leads to the idea that watermark detection and feature extraction can easily be integrated in one application. Both retrieved watermark and hash words can then be sent to a central database for further analysis, to allow identification of content.

In summary, the disclosed method generates robust hashes for multimedia content, for example, audio clips. The audio clip is divided (12) into successive (preferably overlapping) frames. For each frame, the frequency spectrum is divided (15) into bands. A robust property of each band (e.g. energy) is computed (16) and represented (17) by a respective hash bit. An audio clip is thus represented by a concatenation of binary hash words, one for each frame. To identify a possibly compressed audio signal, a block of hash words derived therefrom is matched by a computer (20) with a large database (21). Such matching strategies are also disclosed. In an advantageous embodiment, the extraction process also provides information (19) as to which of the hash bits are the least reliable. Flipping these bits considerably improves the speed and performance of the matching process.